



# OpenGL ES 2.0 : Start Developing Now

Dan Ginsburg



**CONF**  
[www.gdconf.com](http://www.gdconf.com)



# Agenda

- ④ OpenGL ES 2.0  
Brief Overview
- ④ Tools
  - OpenGL ES 2.0 Emulator
  - RenderMonkey w/ OES 2.0 Support
- ④ OpenGL ES 2.0 3D Engine Case Study



# What is OpenGL ES 2.0?

- OpenGL for Embedded Systems
- OpenGL ES 2.0
  - Fully shader-based
  - Based on ES Shading Language
  - Draft spec released at SIGGRAPH 05
  - Spec ratified and released at GDC 07



# OpenGL ES 2.0 – Widespread Industry Support

- OpenGL ES 2.0 support announced from many companies:

AMD

ARM

Imagination Technologies

NVIDIA

STMicro

Texas Instruments

...and more...

- OpenGL ES 2.0 will become ubiquitous

[WWW.GDCONF.COM](http://WWW.GDCONF.COM)




# ES 2.0 – The Problem for Game Developers

- ⊕ Developers need to develop their game engines in advance of new hardware
- ⊕ No hardware available today
- ⊕ OpenGL ES 2.0 may require handheld developers to change their engines significantly
  - Enables more flexibility through programmability



# ES 2.0 – A Development Solution

- ④ OpenGL ES 2.0 Emulator  
OpenGL ES 2.0 implementation for Win32  
Allows developers to write their engines in advance of hardware
- ④ OpenGL ES 2.0 RenderMonkey  
Develop OpenGL ES 2.0 shaders and effects



# OpenGL ES 2.0 Emulator

**CONTROL**  
[www.gdconf.com](http://www.gdconf.com)





# OpenGL ES 2.0 Emulator Goals

- ③ Provide an OpenGL ES 2.0 development environment on the PC
  - Minimize porting effort once hardware is available
- ③ Leverage features/performance of desktop hardware

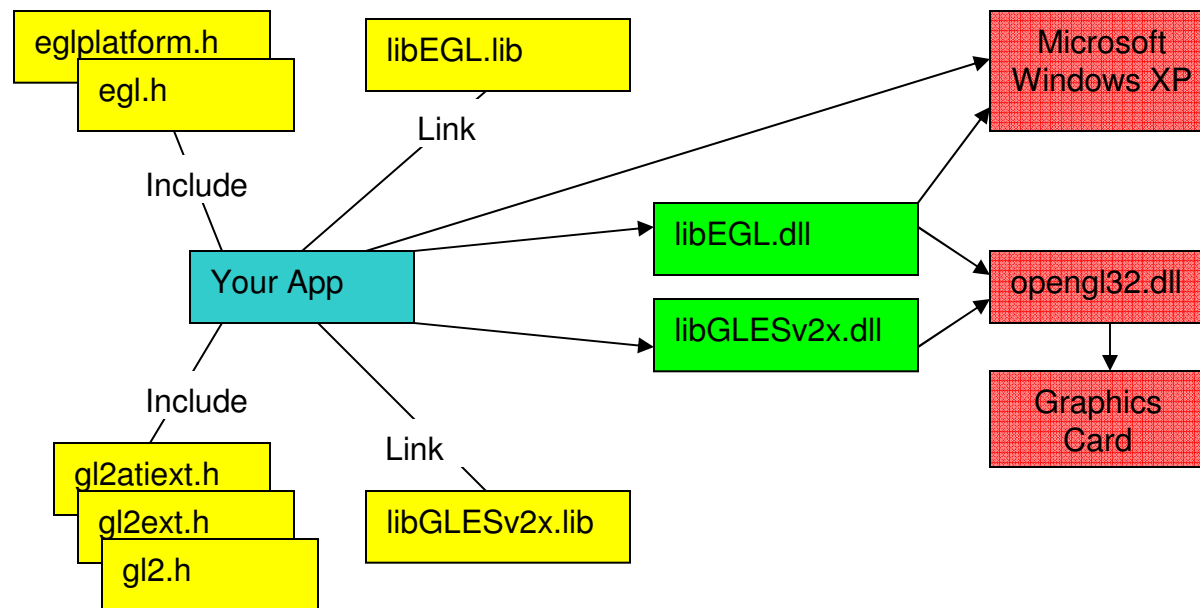




# OpenGL ES 2.0 Emulator – What is it?

- ④ OpenGL ES 2.0 – libGLESv2x.dll + lib
- ④ EGL 1.3 – libEGL.dll + lib
- ④ Khronos standard header files
- ④ Example programs
- ④ Utilizes desktop hardware for rendering  
Requires desktop OpenGL 2.0 hardware

# OpenGL ES 2.0 Emulator – Usage Overview





# OpenGL ES 2.0 Emulator - Features

- ④ OpenGL ES 2.0 Core API
  - ④ Full OpenGL ES 2.0 Implementation
  - ④ Binary and Source Shaders
- ④ Optional Extensions:
  - ④ 10.10.10.2 Vertex/Texture Data
  - ④ FP16 Vertices and Textures
  - ④ 3D and Non-Power-2 Textures
  - ④ Compressed Texture Formats
    - ④ ETC1, ETC3, ETC5, ATI\_TC
  - ④ Occlusion and Conditional Queries
  - ④ Depth Textures



# OpenGL ES 2.0 – Demo



[WWW.GDCONF.COM](http://WWW.GDCONF.COM)



# OpenGL ES 2.0 Emulator – Enables Developers

- ③ More than just a prototyping tool  
Graphics code should move over easily from emulator to real hardware
- ③ Mirrors top tier handheld developer approaches  
Prototype on the PC  
Move to handheld device as a final step



# OpenGL ES 2.0 Emulator

- ③ Contact [devrel@amd.com](mailto:devrel@amd.com) for more information
- ③ PowerVR also provides an emulator and SDK:  
<http://www.powervrinsider.com>





# RenderMonkey – OpenGL ES 2.0 Support

**CONTROL**  
[www.gdconf.com](http://www.gdconf.com)







# RenderMonkey – Why use it?

- ④ Full IDE for shader effect development

- ④ Programmer and artist view for rapid iteration

- ④ Easy integration into game pipeline

- ④ Plug-in SDK for custom import/export

- ④ Effects, models, textures, variables, etc.

- ④ Support for many standard formats

- ④ DDS, BMP, TGA, X, OBJ, 3DS, FX

- ④ Encompasses all effect resources

Render state, texture state, variables, render targets, textures, models, etc...



# RenderMonkey – What's new?

- ③ Support for OpenGL ES 2.0
  - ES Shading Language v1.00
  - ES syntax highlighting
  - ES render/sampler states
  - Large suite of ES examples
  - User editable vertex attribute names

# What is Different with ES Shaders?

- Generic vertex attributes
- User varyings

**PNTTB Stream Mapping**

Usage	Index	Data Type	Attribute Name
POSITION	0	FLOAT3	rm_Vertex
NORMAL	0	FLOAT3	rm_Normal
TEXCOORD	0	FLOAT2	rm_TexCoord0
TANGENT	0	FLOAT3	rm_Tangent
BINORMAL	0	FLOAT3	rm_Binormal

**Shader Editor : Spotlight\_with\_Bump\_OpenGL**

```

uniform mat4 view_proj_matrix;
uniform vec4 view_position;
uniform vec4 lightPos;
uniform vec4 lightDir;

attribute vec4 rm_Vertex;
attribute vec3 rm_Normal;
attribute vec3 rm_Tangent;
attribute vec3 rm_Binormal;
attribute vec3 rm_TexCoord0;

varying vec2 vTexCoord;
varying vec3 vLightVec;
varying vec3 vLightDir;
varying vec3 vViewVec;

varying vec3 vTangent;
varying vec3 vBinormal;
varying vec3 vNormal;

void main(void)
{
    gl_Position = view_proj_matrix * rm_Vertex;

    // construct a tangent space matrix from tan/binorm/normal
    mat3 tanSpace = mat3( rm_Tangent, rm_Normal, rm_Binormal );

    vec3 tangent = vec3( rm_Tangent.x, rm_Tangent.y, rm_Tangent.z );
    vec3 normal = vec3( rm_Normal.x, rm_Normal.y, rm_Normal.z );
    
```

**ES OpenGL ES Preview: Spotlight\_with\_Bump...**



# What is Different with ES Shaders? (cont'd)

- ⌚ Most built-in uniforms removed
  - ⌚ e.g. `gl_ModelViewMatrix`
  - ⌚ RenderMonkey provides equivalent user named uniforms
- ⌚ Default precision qualifier required for FS
- ⌚ Various limitations:
  - Loop constructs
  - Relative addressing
- ⌚ Extension enabling with `#extension`:
  - 3D Textures, derivatives



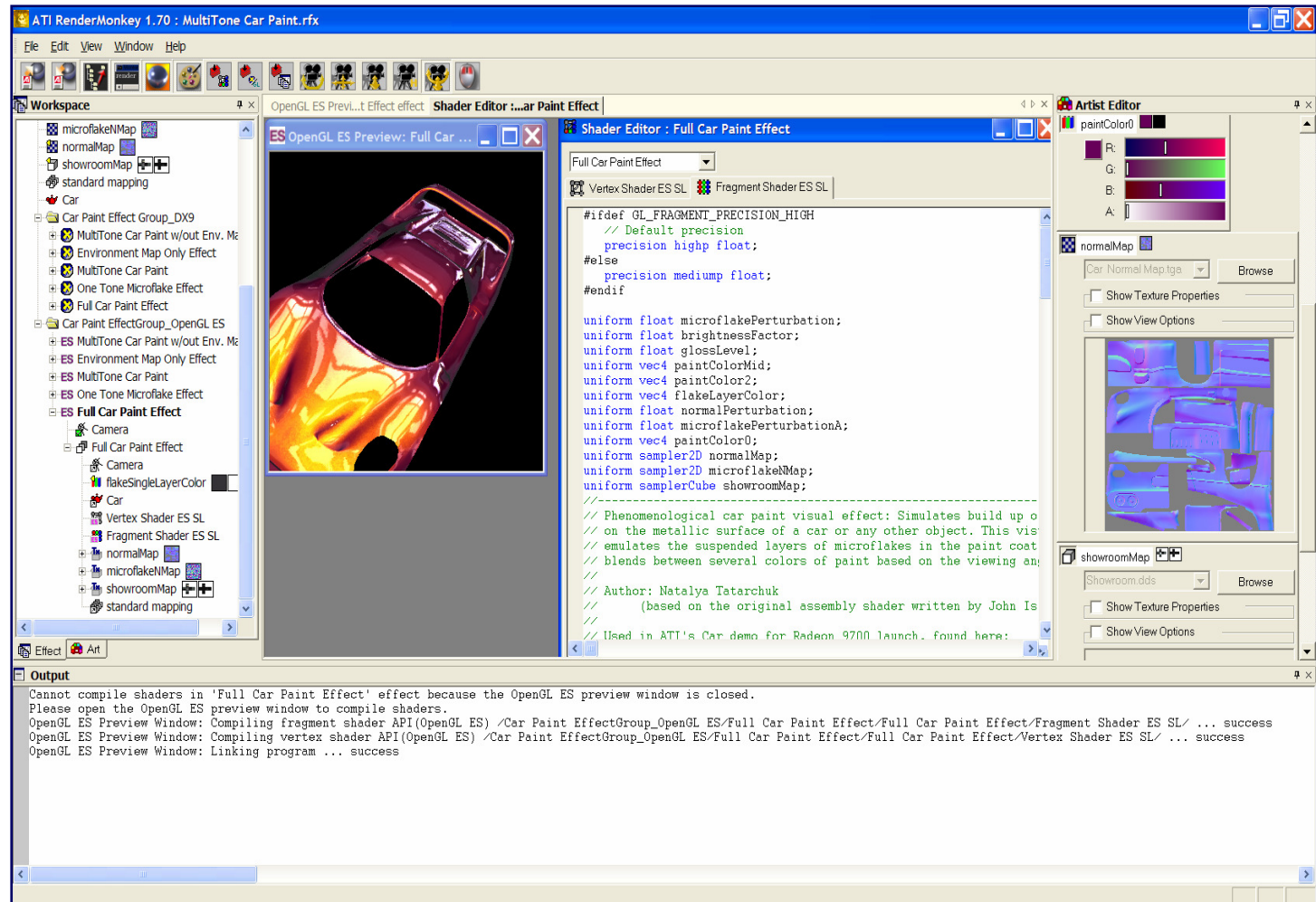
# What is Different with ES Effects? (cont'd)

- ④ Reduced render state
  - ④ Alpha test must be done with *discard*
  - ④ No polygon fill mode
  - ④ No fixed-function state: fog, point size, etc.
- ④ Reduced sampler state
  - ④ Less texture wrap modes
  - ④ No fixed-function LOD bias
  - ④ No texture border color

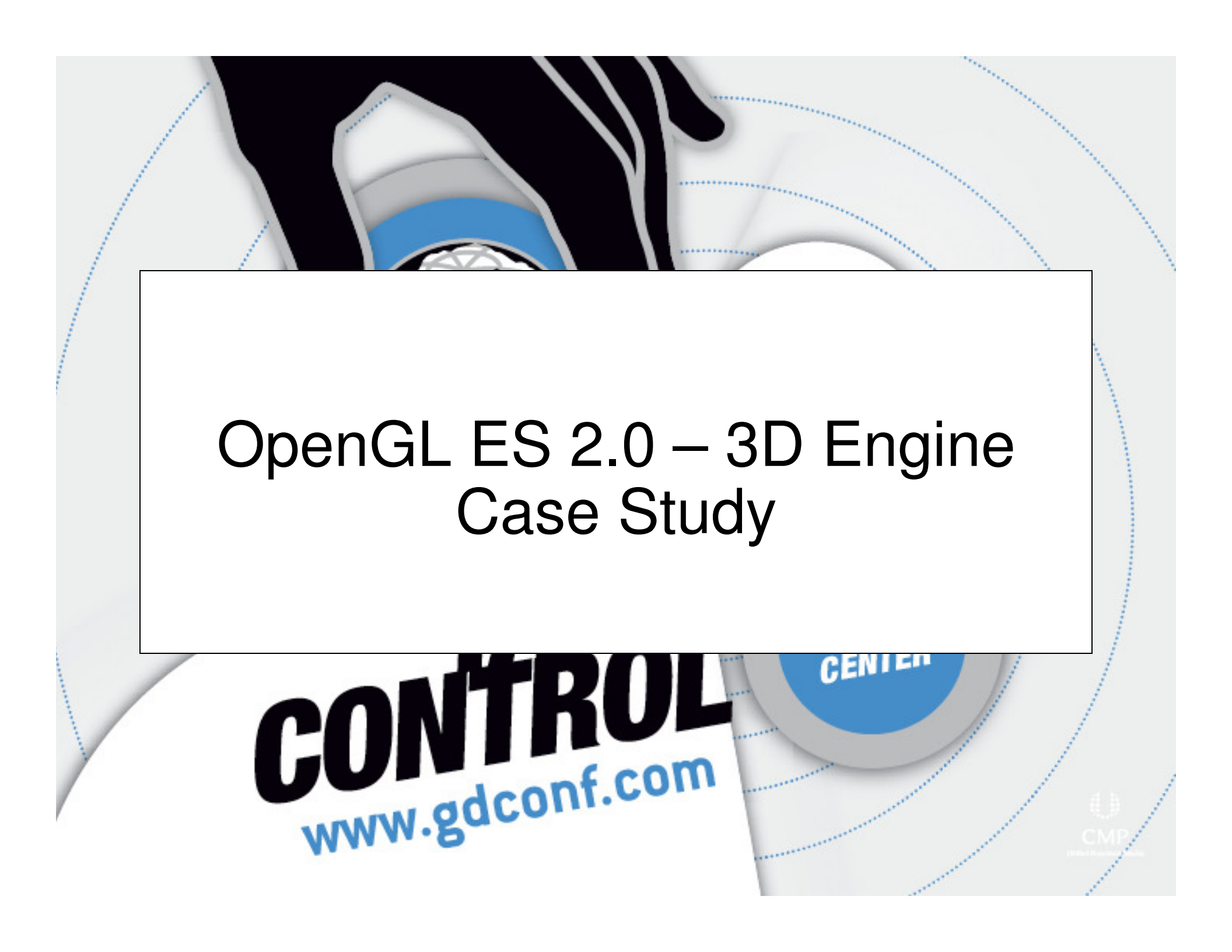




# RenderMonkey - Demo







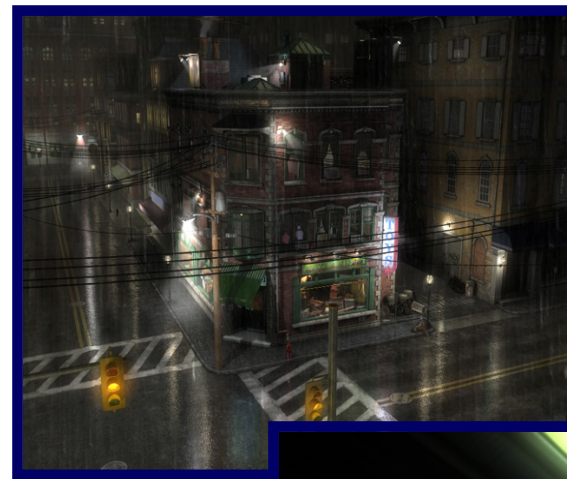
# OpenGL ES 2.0 – 3D Engine Case Study

**CONTROL**  
[www.gdconf.com](http://www.gdconf.com)



# Sushi Demo Engine

- ⊕ AMD's Demo Engine
- ⊕ Support for:
  - DX9
  - DX10
  - OpenGL
  - OpenGL ES 2.0



[WWW.GDCONF.COM](http://WWW.GDCONF.COM)



# Key Challenges

- ⌚ Designing an engine to target multiple APIs with different feature sets
- ⌚ Designing a shader-based engine
- ⌚ Platform compatibility
  - Large variance in handheld platform capabilities
  - Limitations make portability a challenge



# Abstracting the Graphics API

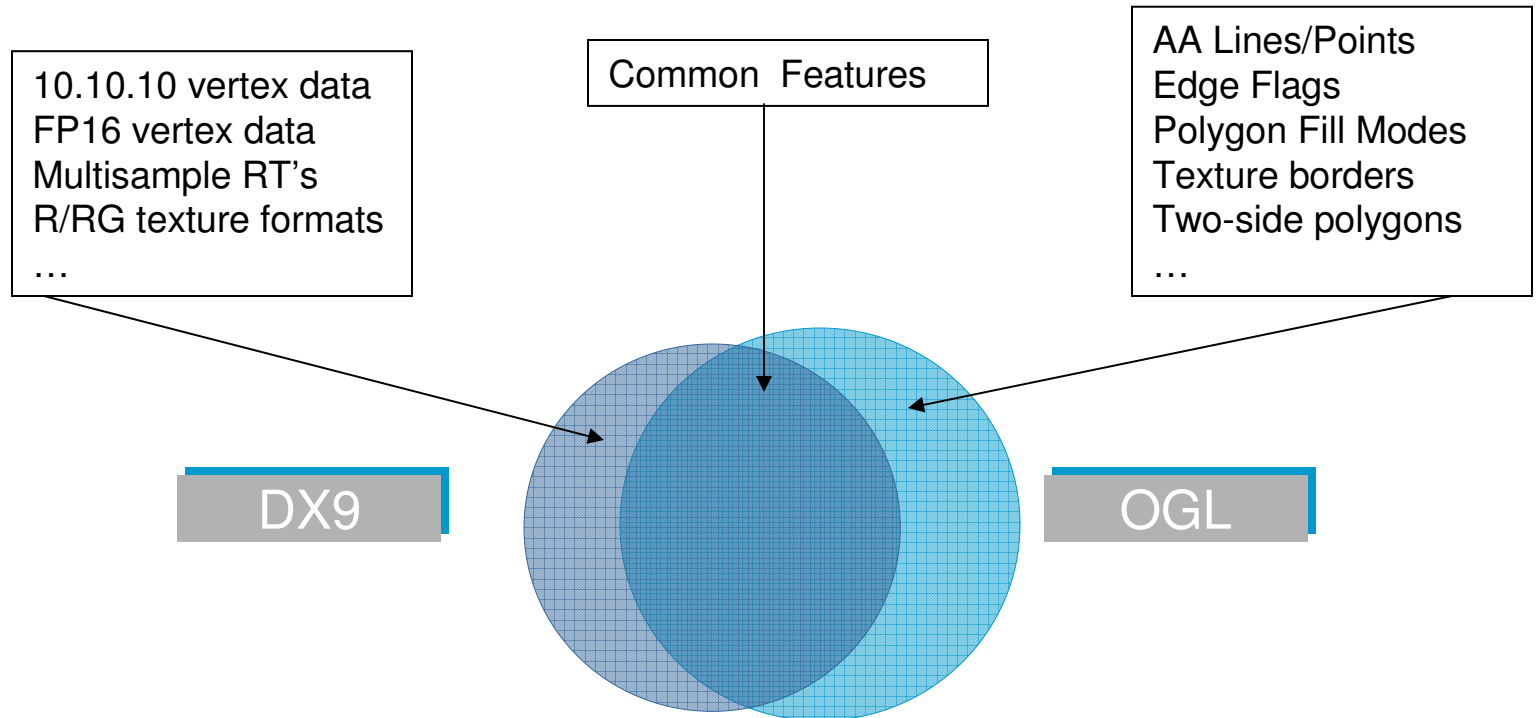
⊕ Challenge: what level to abstract the 3D API?

Support all features of DX9/DX10/GL/GLES?

Support common set of features?

How to handle different shading languages?

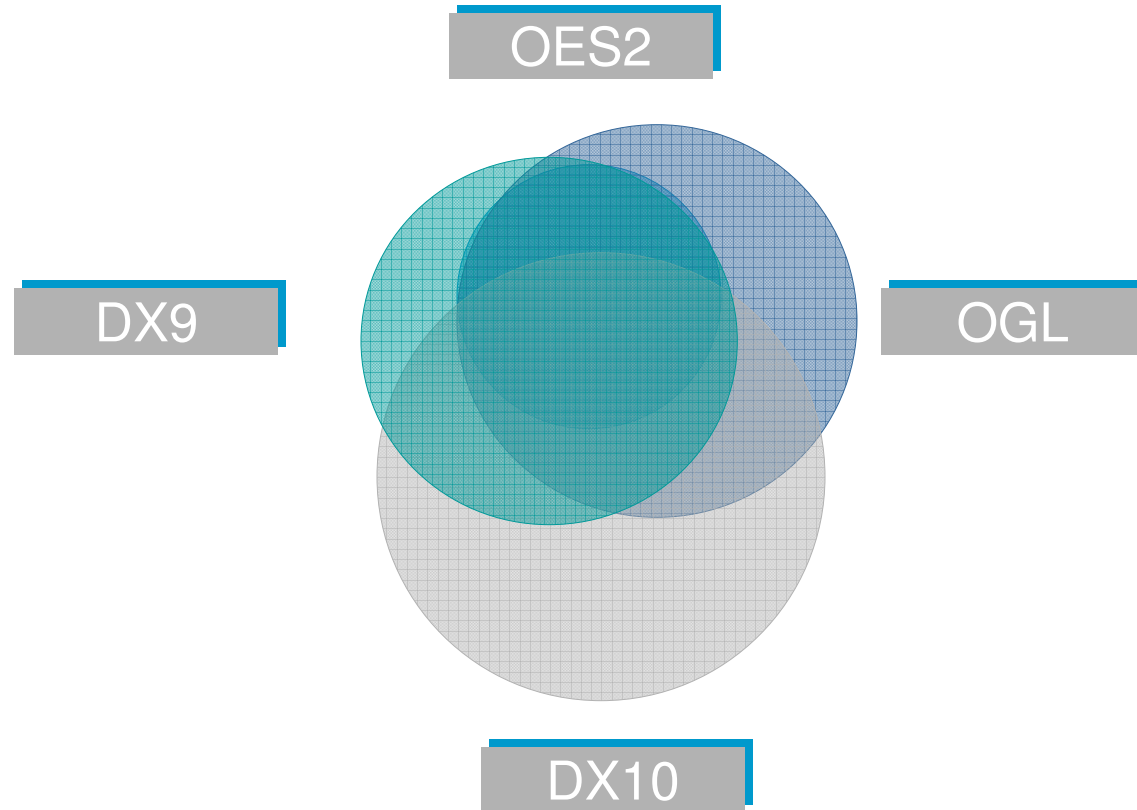
# State of the APIs - 2005



- In 2005, we abstracted the DX9 feature set.
- We used extensions to support missing features in OpenGL.



# State of the APIs - 2007



- The choice is no longer so easy.
- Especially if you add game consoles to the mix...





# Abstracting the API – How We Decided

- ③ Driven by requirements:
  - Demos must use the latest features of all APIs
  - Exposing the lowest-common denominator not an option
- ③ Running the same demo on each API not a requirement
- ③ Let content – rather than API abstraction – define the feature set





# Abstracting the API – What We Did

- ⊕ Our API abstraction looks a lot like DX10
  - Resources
  - Views
  - Geometry Shaders
  - Stream Out
  - All the latest and greatest features...
- ⊕ Each API implementation supports a subset of these features



# API Abstraction – Fallback Paths

- ④ Demo engine is based off a scripting system using Lua
- ④ Lua script provides fallback rendering paths
- ④ Trade off: High end features vs. Content portability

For Sushi, this was a fair tradeoff to make  
It might not be for you...



# Sushi - Effect System

- ④ Encapsulate essential information about rendering techniques
- ④ Essential part of shader-based engine
  - Develop our own?
  - Use someone else's?
    - ④ Microsoft .FX
    - ④ COLLADA FX
    - ④ CgFX
- ④ At the time, no existing solution fully fit our requirements



# Sushi – Effect System Goals

- ④ Multiple API / Shading Language Support  
HLSL, GLSL, ES SL
- ④ Flexible support for advanced rendering techniques

The effect system is the foundation that all the demos are built on



# Sushi Effects – Cross-API Effect System

- Similar to Microsoft .FX, but multiple API support
- Expresses the following data:
  - Shaders
  - Render State
  - Passes
  - Techniques
  - Variable Bindings



# Shader Authoring

- ④ Many of our shaders authored in HLSL
- ④ Needed a way to convert to:
  - OpenGL Shading Language
  - OpenGL ES Shading Language
- ④ Wrote a tool for this purpose:
  - HLSL2GLSL



# HLSL2GLSL

- ④ Command-line tool and library
- ④ Converts SM 3.0 HLSL shaders to:
  - GLSL v1.10.59 shaders
  - ES SL v1.00 shaders
- ④ Open-source:
  - <http://sourceforge.net/projects/hlsl2glsl>
  - Very flexible BSD license





# Sushi – Platform Portability

- ⌘ Handheld platforms have many constraints:

Examples:

- ⌘ No Standard Template Library
- ⌘ No C++ Exceptions
- ⌘ Manual Cleanup Stack
- ⌘ Incomplete Standard Libraries
- ⌘ No Global Variables
- ⌘ Limited Memory Footprint
- ⌘ No Floating Point Unit



# Sushi - Portability

- ③ Standard abstraction layers  
Math, I/O, Memory, Window, etc.
- ③ Custom template classes  
Lists, vectors, maps, etc.
- ③ Constrained use of C++  
No exceptions  
No STL



# Portability - OpenKODE

## ⌕ OpenKODE

Set of portable APIs

Increased source portability

Reduced platform fragmentation

Provisional 1.0 Spec released Feb 2007

- ⌕ Take a look at OpenKODE before designing your own platform abstraction layer...



# Summary

## ⌘ Tools

OpenGL ES 2.0 Emulator

RenderMonkey w/ OES 2.0 Support

## ⌘ OpenGL ES 2.0 3D Engine Case Study

Graphics API Abstraction

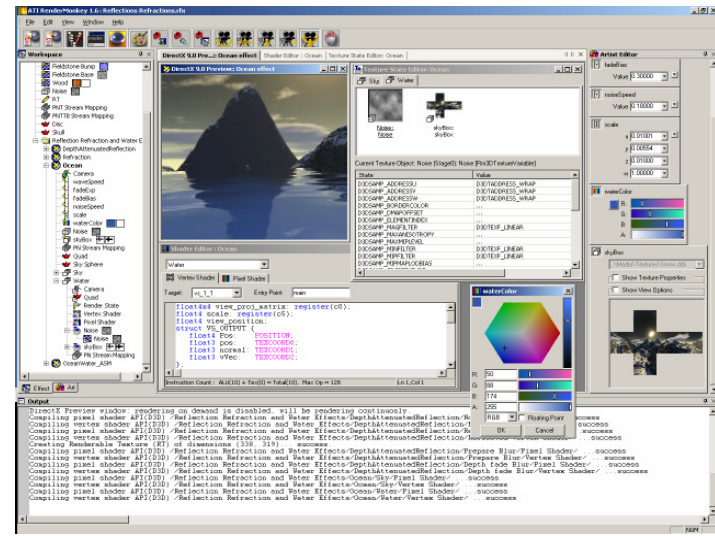
Effects System

Portability



# Questions?

[devrel@amd.com](mailto:devrel@amd.com)



WWW.GDCONF.COM